GROUPS

Group 1 - MM
Ashu Khodwal
Vikas Gupta
Animesh Pal
https://repl.it/join/evxrsjga-assignisik

Group 2 - MM
Soham Banerjee
Sujan Das
Tanmay Santra
https://repl.it/join/hjontikb-assignisik

Group 3 - MM
Luna Biswas
Tarun Borana
Akash Kumar Gupta
Anish Anand
https://repl.it/join/dhtljvlm-assignisik

Group 4 - AB
Uddalok Sarkar
Nidhi Pal
Suraj Yadav
Nikhil Bagaria
https://repl.it/join/trlfjoyl-assignisik

Group 5 - AB
Soham Chatterjee.
Md. Azad Ansari
Awanish Kumar
https://repl.it/join/oqxdxsyg-assignisik

Group 6 - AB
Chhatra Pratap Bharti
Ravindra Sindhiya
Supriya Mandal
Anubhab Trivedi
https://repl.it/join/qhcibrvn-assignisik

Group 7 - MB
Nirupam Basak
Akhil Kumar

Rahul Kushwa
Manideep Aileni
https://repl.it/join/fbmxkcqw-assignisik

Group 8 - MB
Sachin Munda
Ayush Chouhan
Arkadeep Bakshi
Sayan Ray
https://repl.it/join/zcfbcglw-assignisik

PROBLEMS TO SOLVE

Group 1
Q1, Assignment 1, DFS 2018
https://www.isical.ac.in/~dfslab/2018/assignments/assignment1.pdf

Group 2
Q2, Lab Test 1, DFS 2018
https://www.isical.ac.in/~dfslab/2018/exams/lab-test1.pdf

Group 3
Q1, Lab Test 2, DFS 2018
https://www.isical.ac.in/~dfslab/2018/exams/lab-test2.pdf

Group 4
Q3, Lab Test 2, DFS 2018
https://www.isical.ac.in/~dfslab/2018/exams/lab-test2.pdf

Group 5
Q1, Assignment 1, DFS 2019
https://www.isical.ac.in/~dfslab/2019/assignments/assignment1.pdf

Group 6
Q3, Assignment 1, DFS 2019
https://www.isical.ac.in/~dfslab/2019/assignments/assignment1.pdf

Group 7
Q3, Assignment 2, DFS 2019
https://www.isical.ac.in/~dfslab/2019/assignments/assignment2.pdf

Group 8
Q1, Lab Test 1, DFS 2019
https://www.isical.ac.in/~dfslab/2019/exams/lab-test1.pdf

PROGRAMS TO DEBUG

Group 1

Given a string, write a program in C to remove all the duplicates in the given string.

Incorrect code

```
#include <stdio.h>
char *removeDuplicate(char str[], int n){
   int index = 0;
   for (int i=0; i<n; i++){
     int j;
     for (j=0; j<i; j++)
        if (str[i] == str[j])
           break;
     if (j = i)
        str[index++] = str[i];
   }
   return str;
}
int main(){
   char *str= "isimtechcs20-21";
   int n = sizeof(str) / sizeof(str[0]);
   printf("%s", removeDuplicate(str, n));
   return 0;
}
```

Correct code

```
#include <stdio.h>
char *removeDuplicate(char str[], int n){
   int index = 0;
   for (int i=0; i<n; i++){
     int j;
     for (j=0; j<i; j++)
        if (str[i] == str[j])
           break;
     if (j == i)
        str[index++] = str[i];
   }
   return str;
}
```

```
int main(){
   char str[]= "isimtechcs20-21";
   int n = sizeof(str) / sizeof(str[0]);
   printf("%s", removeDuplicate(str, n));
   return 0;
}
```

Group 2

Given a string, write a program in C to find the first non-repeating character in it.

Incorrect code

```
#include <stdio.h>
#include <stdlib.h>
#define NO_OF_CHARS 256
int* getCharCountArray(char* str){
    int* count = (int*)malloc(sizeof(int) * NO_OF_CHARS);
    int i;
    for (i = 0; *(str + i); i++)
        count[*(str + i)]++;
    return count;
}
int firstNonRepeating(char* str){
    int* count = getCharCountArray(str);
    int index = -1, i;
    for (i = 0; *(str + i); i++) {
        if (count[*(str + i)] = 1) {
            index = i;
            break;
        }
    }
    free(count);
    return index;
}
int main(){
    char str[] = "isimtechcs20-21";
    int index = firstNonRepeating(str);
    if (index == -1)
        printf("All characters are repeating / String is empty");
    else
        printf("First non-repeating character: %c", str[index]);
    getchar();
    return 0;
}
```

Correct code

```
#include <stdio.h>
#include <stdlib.h>
#define NO_OF_CHARS 256
int* getCharCountArray(char* str){
    int* count = (int*)calloc(sizeof(int), NO_OF_CHARS);
    int i;
    for (i = 0; *(str + i); i++)
        count[*(str + i)]++;
    return count;
}
int firstNonRepeating(char* str){
    int* count = getCharCountArray(str);
    int index = -1, i;
    for (i = 0; *(str + i); i++) {
        if (count[*(str + i)] == 1) {
            index = i;
            break;
        }
    }
    free(count);
    return index;
}
int main(){
    char str[] = "isimtechcs20-21";
    int index = firstNonRepeating(str);
    if (index == -1)
        printf("All characters are repeating / String is empty");
    else
        printf("First non-repeating character: %c", str[index]);
    getchar();
    return 0;
}
```

Group 3

Write a recursive function in C to print the reverse of a given string.

Incorrect code

```
# include <string.h>
void reverse(char *str){
  if (*str == '\0'){
```

```
        reverse(str+1);
        printf("%c", *str);
    }
}
int main(){
    char *a = "malayalam";
    reverse(&a);
    return 0;
}
```

Correct code

```
# include <stdio.h>
void reverse(char *str){
    if (*str){
        reverse(str+1);
        printf("%c", *str);
    }
}
int main(){
    char a[] = "malayalam";
    reverse(a);
    return 0;
}
```

Group 4

Write a program in C to reverse an array.

Incorrect code

```
#include <stdio.h>
void swap(int* a, int* b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
void reverse(int array[], int array_size){
    int *pointer1 = array, *pointer2 = array + array_size - 1;
    while (pointer1 < pointer2){
        swap(&pointer1, &pointer2);
        pointer1++;
        pointer2++;
    }
}
void print(int* array, int array_size){
```

```
    int *length = array + array_size, *position = array;
    for(position = array; position <= length; position++)
        printf("%d ", position);
}
int main(){
    int array[] = {2, 4, -6, 5, 8, -1};
    printf("\nOriginal array: \n");
    printf(array, 6);
    printf("\nReversed array: \n");
    reverse(array, 6);
    printf(array, 6);
    return 0;
}
```

Correct code

```
#include <stdio.h>
void swap(int* a, int* b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
void reverse(int array[], int array_size){
    int *pointer1 = array, *pointer2 = array + array_size - 1;
    while (pointer1 < pointer2){
        swap(pointer1, pointer2);
        pointer1++;
        pointer2--;
    }
}
void print(int* array, int array_size){
    int *length = array + array_size, *position = array;
    for(position = array; position < length; position++)
        printf("%d ", *position);
}
int main(){
    int array[] = {2, 4, -6, 5, 8, -1};
    printf("\nOriginal array: \n");
    print(array, 6);
    printf("\nReversed array: \n");
    reverse(array, 6);
    print(array, 6);
    return 0;
}
```

Group 5

Write a program in C to find out the multiplication result of two 3x3 matrices.

Incorrect code

```
#include <stdio.h>
#define ROW 3
#define COL 3
void matrixInput(int mat[][]);
void matrixPrint(int mat[][]);
void matrixMultiply(int mat1[][], int mat2[][], int res[][]);
int main(){
    int mat1[ROW][COL];
    int mat2[ROW][COL];
    int product[ROW][COL];
    printf("Enter elements in first matrix of size %dx%d\n", ROW, COL);
    matrixInput(mat1);
    printf("Enter elements in second matrix of size %dx%d\n", ROW, COL);
    matrixInput(mat2);
    matrixMultiply(mat1, mat2, product);
    printf("Product of both matrices is : \n");
    matrixPrint(product);
    return 0;
}
void matrixInput(int mat[][COL]){
    int row, col;
    for (row = 0; row < ROW; row++)
        for (col = 0; col < COL; col++);
            scanf("%d", (*(mat + row) + col));
}
void matrixPrint(int *mat[][COL]){
    int row, col;
    for (row = 0; row < ROW; row++){
        for (col = 0; col < COL; col++)
            printf("%d ", *(*(mat + row) + col));
        printf("\n");
    }
}
void matrixMultiply(int mat1[][COL], int mat2[][COL], int res[][COL]){
    int row, col, i;
    int sum;
    for (row = 0; row < ROW; row++){
        for (col = 0; col < COL; col++){
```

```
            sum = 0;
            for (i = 0; i < COL; i++)
                sum += (**(mat1 + row) + i) * (**(mat2 + i) + col);
            *(*(res + row) + col) = sum;
        }
    }
}
```

Correct code

```
#include <stdio.h>
#define ROW 3
#define COL 3
void matrixInput(int mat[][COL]);
void matrixPrint(int mat[][COL]);
void matrixMultiply(int mat1[][COL], int mat2[][COL], int res[][COL]);
int main(){
    int mat1[ROW][COL];
    int mat2[ROW][COL];
    int product[ROW][COL];
    printf("Enter elements in first matrix of size %dx%d\n", ROW, COL);
    matrixInput(mat1);
    printf("Enter elements in second matrix of size %dx%d\n", ROW, COL);
    matrixInput(mat2);
    matrixMultiply(mat1, mat2, product);
    printf("Product of both matrices is : \n");
    matrixPrint(product);
    return 0;
}
void matrixInput(int mat[][COL]){
    int row, col;
    for (row = 0; row < ROW; row++)
        for (col = 0; col < COL; col++)
            scanf("%d", (*(mat + row) + col));
}
void matrixPrint(int mat[][COL]){
    int row, col;
    for (row = 0; row < ROW; row++){
        for (col = 0; col < COL; col++)
            printf("%d ", *(*(mat + row) + col));
        printf("\n");
    }
}
void matrixMultiply(int mat1[][COL], int mat2[][COL], int res[][COL]){
```

```
    int row, col, i;
    int sum;
    for (row = 0; row < ROW; row++){
        for (col = 0; col < COL; col++){
            sum = 0;
            for (i = 0; i < COL; i++)
                sum += (*(*(mat1 + row) + i)) * (*(*(mat2 + i) + col));
            *(*(res + row) + col) = sum;
        }
    }
}
```

Group 6

Given two sorted arrays, write a program in C to find their intersection.

Incorrect code

```
#include <stdio.h>
int printIntersection(int arr1, int arr2, int m, int n){
    int int = 0, j = 0;
    while (i < m && j < n){
        if (arr1[i] < arr2[j])
            i++;
        else if (arr2[j] < arr1[i])
            j++;
        else{
            print(" %d ", arr2[i++]);
            j++;
        }
    }
}
int main(){
    int arr1[] = { 1, 2, 4, 5, 6 };
    int arr2[] = { 2, 3, 5, 7 };
    int m = sizeof(*arr1) / sizeof(arr1[0]);
    int n = sizeof(*arr2) / sizeof(arr2[0]);
    printIntersection(arr1, arr2, m, n);
    return 0;
}
```

Correct code

```
#include <stdio.h>
int printIntersection(int arr1[], int arr2[], int m, int n){
```

```
    int i = 0, j = 0;
    while (i < m && j < n){
        if (arr1[i] < arr2[j])
            i++;
        else if (arr2[j] < arr1[i])
            j++;
        else{
            printf(" %d ", arr2[j++]);
            i++;
        }
    }
}
int main(){
    int arr1[] = { 1, 2, 4, 5, 6 };
    int arr2[] = { 2, 3, 5, 7 };
    int m = sizeof(arr1) / sizeof(arr1[0]);
    int n = sizeof(arr2) / sizeof(arr2[0]);
    printIntersection(arr1, arr2, m, n);
    return 0;
}
```

Group 7

Given two sorted arrays, write a program in C to find their union.

Incorrect code

```
#include <stdio.h>
int printUnion(int arr1[], int arr2[], int m, int n){
    int i = 0, j = 0;
    while (i < m & j < n){
        if (arr1[i] < arr2[j])
            printf(" %d ", arr1[i++]);
        else if (arr2[j] < arr1[i])
            printf(" %d ", arr2[j++]);
        else{
            printf(" %d ", arr2[j++]);
            i++;
        }
    }
    while (j < n)
        printf(" %d ", arr2[j++]);
    while (i < m)
        printf(" %d ", arr1[i++]);
}
```

```
int main(){
    int arr1[] = { 1, 2, 4, 5, 6 };
    int arr2[] = { 2, 3, 5, 7 };
    int m = sizeof(*arr1) / sizeof(arr1[0]);
    int n = sizeof(*arr2) / sizeof(arr2[0]);
    printUnion(arr1, arr2, m, n);
    return 0;
}
```

Correct code

```
#include <stdio.h>
int printUnion(int arr1[], int arr2[], int m, int n){
    int i = 0, j = 0;
    while (i < m && j < n){
        if (arr1[i] < arr2[j])
            printf(" %d ", arr1[i++]);
        else if (arr2[j] < arr1[i])
            printf(" %d ", arr2[j++]);
        else{
            printf(" %d ", arr2[j++]);
            i++;
        }
    }
    while (i < m)
        printf(" %d ", arr1[i++]);
    while (j < n)
        printf(" %d ", arr2[j++]);
}
int main(){
    int arr1[] = { 1, 2, 4, 5, 6 };
    int arr2[] = { 2, 3, 5, 7 };
    int m = sizeof(arr1) / sizeof(arr1[0]);
    int n = sizeof(arr2) / sizeof(arr2[0]);
    printUnion(arr1, arr2, m, n);
    return 0;
}
```

Group 8

Given an array of integers, write a program in C to find the sum of its elements.

Incorrect code

```
#include <stdlib.h>
```

```
int sum(int arr[], int n){
    int sum = 0;
    for (int i = 0; i <= n; i++)
    sum += arr[i];
    return sum;
}
int main(){
    int arr[] = {12, 3, 4, 15};
    int n = sizeof(*arr) / sizeof(arr[0]);
    printf("Sum of the given array is: %d", sum(arr, n));
    return 0;
}
```

Correct code

```
#include <stdio.h>
int sum(int arr[], int n){
    int sum = 0;
    for (int i = 0; i < n; i++)
    sum += arr[i];
    return sum;
}
int main(){
    int arr[] = {12, 3, 4, 15};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Sum of the given array is: %d", sum(arr, n));
    return 0;
}
```

TOOLS

https://repl.it